Constrained geodesic trajectory generation on learnt skill manifolds

Ioannis Havoutis Subramanian Ramamoorthy
School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK
I.Havoutis@sms.ed.ac.uk, S.Ramamoorthy@ed.ac.uk

***Abstract*— This paper addresses the problem of compactly encoding a continuous family of trajectories corresponding to a robotic skill, and using this representation for the purpose of *constrained* trajectory generation in an environment with many (possibly dynamic) obstacles. With a skill manifold that is learnt from data, we show that constraints can be naturally handled within an iterative process of minimizing the total geodesic path length and curvature over the manifold. We demonstrate the utility of this process with two examples. Firstly, a three-link arm whose joint space and corresponding skill manifold can be explicitly visualized. Then, we demonstrate how this procedure can be used to generate constrained walking motions in a humanoid robot.**

I. Introduction

Humanoid robots receive increasing attention as general purpose platforms suitable to a multitude of applications. However, the level of flexibility that can actually be achieved tends to fall short of this promise of generic dexterity. One of the big difficulties is related to the problem of devising motion planning and control algorithms that can cope with the combination of dynamic complexity, dimensionality and model imprecision. Many off the shelf solutions providing humanoid behaviours, e.g., for locomotion, tend to be restricted to a limited and discrete vocabulary, valid only in narrow domains of applicability. For instance, it is hard to find a general purpose humanoid walking 'engine' that provides full control over step length, width and height, in real-world terrains. On the other end, specialized approaches that do enable some flexible movements tend to be computationally expensive, e.g., requiring high-dimensional numerical optimization and/or c-space search, often with near-exact knowledge of the system and its environment. This is a steep requirement for resource constrained machines.

In this paper, we address the problem of designing motion strategies that can achieve a rich set of within-skill variations. These strategies are acquired in a data-driven manner, allowing for adaptation to changes in environmental conditions and task contexts, and can be implemented in realistic resource constrained robotic systems. Such a representation of a parameterized skill, applicable under a wide variety of conditions, could then form the basis for higher level search processes over a small alphabet, enabling fast high level planning (e.g., [1]). Indeed, one of the big weaknesses of some existing motion synthesis strategies is that they do not cleanly admit such abstractions.

We build on earlier work [2] – to compactly represent a continuous family of trajectories representing a specific skill such as variable step-length walking - to incorporate constraints (involving a combination of task and joint space obstacles). The goal is to define a scheme wherein the manifold captures the essential variations in the set of trajectories corresponding to a skill, from which one is able to lazily select specific instances as the constraints (possibly dynamic) are revealed. In practice, one often adopts a receding-horizon approach to handling such problems and we'd like our approach to be compatible with this paradigm.

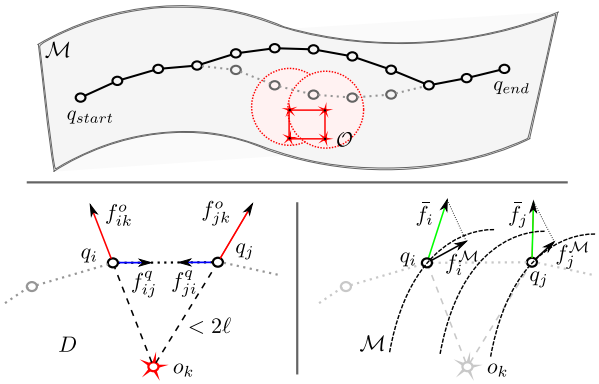


Fig. 1. *Top*; Sketch of the constrained optimization procedure, where the obstacle set $\mathcal{O}$ drives the trajectory away from the red square obstacle. *Left*; An obstacle point o_k affects only the path points that are within its range (2ℓ) and exerts on them repulsive forces (red). In contrast the path points are modeled as a spring system and points of the path can exert repulsive (not shown) and attractive forces (blue) to their path neighbors. *Right*; All forces that act on each path point are averaged and the resulting mean vector is subsequently projected on the learnt manifold $\mathcal{M}$.

The basic notion of utilizing low-dimensional representations in a motion synthesis setting is becoming well accepted in the robotics and graphics communities [3], [4], [5]. Some recent works [6], [7], [8] address this issue by considering how task space constraints, e.g., end-effector constraints, can be used to structure planning in configuration space with local Jacobian mappings. However, this requires full access to an exact model, which may not always be possible. The machine learning literature includes many examples of dimensionality reduction methods used to abstract and/or make problem spaces manageable. Wang et al. [9] introduced the GPDM, which identifies a mapping to a low dimensional space where a linear dynamic model is fit to data. In the same spirit, Bitzer et al. [10] use a Gaussian Process-based nonlinear dimensionality reduction technique to arrive at a subspace within which one may approximate demonstrated data using parameterized families of paths. An approach that is closer to our current work is that of Calinon et al. [11], who demonstrate robot programming by demonstration with a probabilistic model, namely Gaussian Mixture Regression, of Jacobian-based inverse kinematics for learning trajectories and incorporating task space constraints. What has not always been exploited in such work is the *geometrical structure* of families of paths in lower dimensional subspaces. By simply mapping a set of poses to a low-dimensional space and fitting a parameterized model, one is essentially *overriding* potential intrinsic dynamics effects that define many behaviours of interest.

Our goal is to learn this geometric structure, i.e., a skill manifold, that captures the intrinsic structure of the space of trajectories by approximating the tangent space from demonstration data. So, if one begins with a set of motion examples

from a specific class, e.g., due to a path optimization or redundancy resolution principle or even a more complex kinodynamic constraint, then one seeks a representation that intrinsically captures both the restriction of states to a low-dimensional space *and* the evolution of the trajectories in that space - as opposed to imposing a trajectory generation scheme, *post hoc*.

II. Manifold learning

Our nonlinear manifold learning method is based on Locally Smooth Manifold Learning by Dollar et al. [12], which we have adapted with robot motion-specific issues in mind. In particular, we replace the neighborhood graph creation process with a procedure that considers task space distances and the need to ensure that temporal neighborhood relations along the demonstrated trajectories are respected, similar to the procedure used in ST-Isomap [13].

In the usual formulation, manifold learning is aimed at finding an embedding or 'unrolling' of a nonlinear manifold onto a lower dimensional space while preserving metric properties such as inter-point distances. Much of this work, e.g., *Isomap*, *LLE*, has been focused on summarization, visualization or analysis that explains some aspect of the observed data. On the other hand, we are interested in preserving properties of trajectories in the data set. So, our goal is to learn a model of the tangent space of the low-dimensional nonlinear manifold, conditioned on the adjacency relations of the high dimensional data. Such a learnt manifold model can then be used to compute geodesic distances, to find projections of points on the manifold and to directly generate geodesic *paths* between points.

A. Learning the model

Given that our D-dimensional data lies on a locally smooth d-dimensional manifold in D-dimensional space, where $d < D$, there exists a continuous bijective mapping $\mathcal{M}$ that converts low dimensional points $y \in \mathbb{R}^d$ from the manifold, to points $x \in \mathbb{R}^D$ of the high dimensional space, $x = \mathcal{M}(y)$. The goal is to learn a mapping from a point on the manifold to its tangent basis $\mathcal{H}(x)$,

$$\mathcal{H}: \; x \in \mathbb{R}^D \mapsto \left[\frac{\partial}{\partial y_1}\mathcal{M}(y) \cdots \frac{\partial}{\partial y_d}\mathcal{M}(y)\right] \in \mathbb{R}^{D\times d}$$

where each column of $\mathcal{H}(x)$ is a basis vector of the tangent space of the manifold at y, i.e. the partial derivative of $\mathcal{M}$ with respect to y.

Learning a model of the mapping with some parametrization θ, i.e. $\mathcal{H}_\theta$, is done as follows. Given two neighboring points on the manifold, x^i and x^j, the difference between these points, $\Delta^i_{.j}$, should be a linear combination of the tangent vectors at that point on the manifold, scaled by an unknown alignment factor. Taking $\Delta^i_{.j}$ to be the centered estimate of the directional derivative at $\bar{x}^{ij}$ and ϵ^{ij} to be the unknown alignment factor, we have $\mathcal{H}_\theta(\bar{x}^{ij})\epsilon^{ij} \approx \Delta^i_{.j}$, that holds given ϵ is small enough and the manifold can be locally approximated with a quadratic form. To learn $\mathcal{H}_\theta$ we define the error function:

$$err(\theta) = \min_{\{\epsilon^{ij}\}} \sum_{i,j\in N^i} \left\|\mathcal{H}_\theta(\bar{x}^{ij})\epsilon^{ij} - \Delta^i_{.j}\right\|_2^2,$$

where N^i is the set of neighbors of x^i. This minimization problem for θ is solved with a regularization term that ensures that the ϵ's do not get too large, that the tangents do not get too small and that neighboring tangent bases are aligned. For a precise model of the tangent space one would need to compute the tangent basis for each point, $\mathcal{H}_\theta(\bar{x}^{ij})$, which can be considered as a regression over the evidence (training data), and compute the alignment factors, ϵ^{ij}, for all neighboring points. Solving for the bases and their alignment simultaneously is complex, but if either one is kept constant, solving for the remaining variables becomes a tractable least squares problem.

Modeling $\mathcal{H}_\theta$ is done with a linear model of radial basis functions (RBF's) with features over the evidence [14], where the number of basis functions, f, acts as parameter that can control the smoothness of the estimated mapping. More nonsmooth nonlinear manifolds with abrupt changes, would typically require more basis functions to ensure a tight local fit, though the generalization ability may be weakened. Optimizing the model requires alternating between the two least squares problems described above, until a local minima has been reached. Typically more than one random restart is performed to avoid bad local minima.

B. Optimal geodesic paths

By approximating the tangent space of the manifold, we gain access to a variety of geometric operations. Central to our robotics aims is the ability to compute *paths* through configuration space that lie on the low dimensional manifold. In this spirit, we now change our notation of points from x to q, to denote poses a robot can achieve in a configuration space.

Our goal is to find the shortest path between two specified poses q_{start} and $q_{end} \in \mathbb{R}^D$, D being the dimensionality of the configuration space, that respects the geometry of the learnt manifold. In a robotics context, being on the manifold essentially means that the constraints (e.g., optimality w.r.t. a particular task-specific cost) inherent in the training data are satisfied. In practice, we discretize our path into a set of n via points, $\mathbf{q} = q_{start}, \ldots, q_{end}$, with q_{start} and q_{end} being fixed, and we follow a combination of gradient descent steps to minimize the length of the path while not leaving the support of the manifold.

The initial estimate of the shortest path is computed by interpolating linearly between q_{start} and q_{end}, while following the geometry of the manifold, until the distance between consecutive points is acceptable. Since we have learnt the tangent space of the manifold we can find a minimum energy solution that follows the orthonormal (to the manifold) component of the gradient of

$$err_{\mathcal{M}}(\mathbf{q}) = \min_{\{\epsilon^{ij}\}} \sum_{i,j\in N^i} \left\|\mathcal{H}_\theta(\bar{q}^{ij})\epsilon^{ij} - (q^i - q^j)\right\|_2^2,$$

that essentially makes the q^is "stick" to the learnt manifold by iteratively moving them to points where neighboring (consecutive) bases are aligned. Next we apply another gradient descent optimization by following the parallel (to the manifold) component of

$$err_{length}(\mathbf{q}) = \sum_{i=2}^{n} \left\|q^i - q^{i-1}\right\|_2^2,$$

that iteratively minimizes the length of the path without leaving the support of the learnt manifold, while keeping the endpoints fixed.

C. Constrained geodesic paths

In practice, we often require more control over the generated trajectories as, often, the system would need to avoid

task space and joint space obstacles. This is a constrained trajectory generation problem over the manifold. We now describe a procedure for generating constrained geodesic paths that avoid *"no-go"* patches on the manifold surface. These are defined as sets of obstacle points $\mathcal{O} \in \mathbb{R}^D$ that are uniformly sampled from the *"no-go"* task space region and trace the obstacle patch in joint space. For example such points can be samples from the faces of a cube obstacle or a set of points sampled from the surface of a sphere.

The intersection of the manifold set and the obstacle set, $\mathcal{M} \cap \mathcal{O}$, is the region that we would like to take into consideration when generating a constrained geodesic path. This point set would drive the geodesic path away from the patch that we want to avoid but given the learnt tangent space the path will not leave the surface of the manifold, thus the optimality properties that the manifold represents.

We treat the affected consecutive geodesic path points, $\mathbf{q}$, as a system of springs that can either exert attractive or repulsive forces to their neighbors. A force, f_{ij}^q, between two consecutive path points q_i and q_j, is repulsive if the distance between them is less than ℓ, and attractive if the distance is greater than ℓ. The distance ℓ is a metric that is derived from the manifold geometry and is the spacing between consecutive points, while the strength of the forces is dependent to the distance difference.

The obstacle point set, $\mathcal{O}$, exerts repulsive forces to the path points. The area of effect of the obstacle point set is also defined relative to ℓ; each obstacle point, o_k exerts a force, f_{ik}^o, to every path point, q_i, within a hypersphere of radius set to 2ℓ. This distance can also be increased or decreased with obstacle clearance in mind.

We calculate all forces that act on each affected path point and compute a mean force vector. This vector is projected on the manifold and each point is moved by a small step accordingly. We repeat the procedure until all points have cleared obstacle points or the algorithm has converged. Fig. 1 provides a sketch of the procedure.

Algorithm 1 Constrained Geodesic Trajectory Generation

INPUT: $\mathcal{M},\ q_{start},\ q_{end},\ \mathcal{O}$
OUTPUT: $\mathbf{q} \equiv \{q_{start}, \ldots, q_i, q_{end}\}$
$\mathbf{q} \leftarrow$ Optimal Geodesic Path($q_{start},\ q_{end}$)
repeat
  $d^{\mathcal{O}} \leftarrow$ Compute Distances($\mathbf{q},\ \mathcal{O}$)
  $[f_{ik}^o,\ f_{ij}^q] \leftarrow$ Calculate Spring Forces($\mathbf{q},\ d^{\mathcal{O}}$)
  $\bar{f}_i \leftarrow f_{i.}^o + f_{i.}^q$
  $\bar{f}_i^{\mathcal{M}} \leftarrow \bar{f}_i \mathcal{H}_\theta^{\mathbf{q}} \mathcal{H}_\theta^{\mathbf{q}\prime}$
  $\mathbf{q} \leftarrow \mathbf{q} + \gamma \bar{f}_i^{\mathcal{M}}$
  $\mathcal{C}^i \leftarrow \partial \mathbf{q}^2 / \partial s^2$ {Curvature}
  $\bar{\mathcal{C}} \leftarrow 1/n \sum \mathcal{C}^i$
  $\bar{\mathcal{C}}_i^{\mathcal{M}} \leftarrow (\bar{\mathcal{C}} - \mathcal{C}^i) \mathcal{H}_\theta^{\mathbf{q}} \mathcal{H}_\theta^{\mathbf{q}\prime}$
  $\mathbf{q} \leftarrow \mathbf{q} + \gamma \bar{\mathcal{C}}_i^{\mathcal{M}}$
  $\delta \leftarrow \mathbf{q} - \mathbf{q}_{old}$
until $d^{\mathcal{O}} = null$ *or* $\delta \leq 10^{-6}$

1) Curvature smoothing: The above procedure only acts on the geodesic path points that are in the area of effect (distance $< 2\ell$) of obstacle points. This tends to lead to trajectories that are not smooth when only small portions of the paths are near obstacles. To alleviate this, we introduce a step that considers the full set of path points and interplays with the constraint optimization.

We use the path curvature as a measure for smoothing the generated geodesic paths in a structured fashion. We calculate the curvature, $\mathcal{C}$, over the discrete geodesic points, $\mathbf{q} = q_1, \ldots, q_n$ as

$$\mathcal{C}^i = \frac{\partial \mathbf{q}^2}{\partial s^2},\ i = 1, \ldots, n-2,$$

where s is the distance between two consecutive points. We calculate the mean curvature $\bar{\mathcal{C}}$ and the error gradient $\bar{\mathcal{C}} - \mathcal{C}^i$ (vector), for each triple of path points. Each point is then moved by a small step along the error gradient and projected on the manifold tangent space. The entire procedure is summarized in Alg. 1. The following sections present two examples of our method. The first example presents experiments on a simulated 3-link arm where both the manifold and the learnt model can be visualized and are representative of the core ideas behind this work. For the second example we use a physical humanoid robot, with which we demonstrate how our method scales to more complex systems and more challenging tasks.

III. Experiments on a robotic arm

Our first set of experiments are designed to elucidate the basic concepts underlying our approach. We use a *3-link planar* arm where we can explicitly visualize both the configuration space and the optimization manifold (surface that corresponds to a specific redundancy resolution strategy), along with possible obstacle points. The arm is a series of three rigid links, of $1/3$ length, that are coupled with hinge joints, producing a redundant system with 3 degrees of freedom (DoFs) that is constrained to move on a 2 dimensional plane (task space).

A. Training data

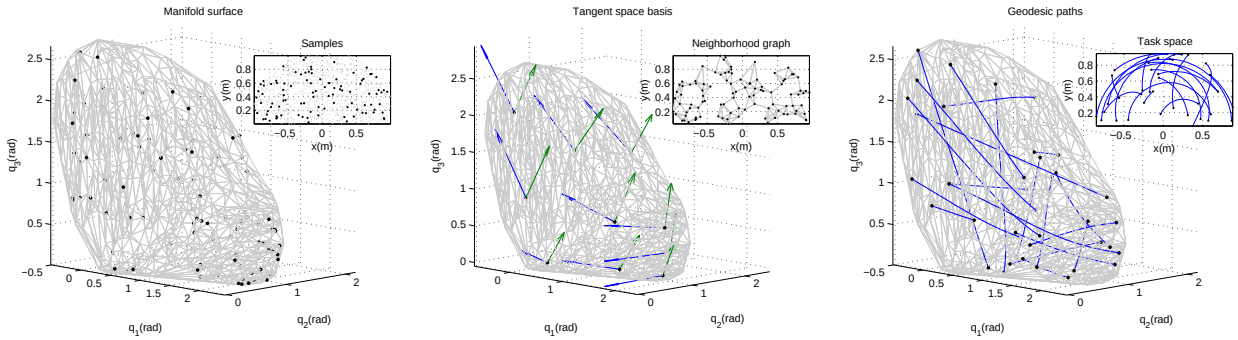
We randomly sample 100 Cartesian points from the top semicircle of the task space of the system. The dataset is 100 points of x and y couples, where $-1 \leq x \leq 1$ and $0 \leq y \leq 1$. We run the task space dataset through an iterative optimization procedure detailed below and get the corresponding joint space datapoints, $\mathbf{q} = (q_1, q_2, q_3)$. A set of 100 such points is depicted in Fig. 2(a), as black dots in joint space and task space plots. We densely sample the space with 900 more points that are used solely for visualization purposes and play no further part in the learning procedure. For visualization purposes, we use all 1000 points to compute a Delaunay triangulation of the joint space structure as sampled, and then plot the trimesh (triangle mesh) for comparison with the paths that our algorithm produces. This trimesh surface is depicted in all figures with thin gray edges.

The system being redundant, we first have to choose a redundancy resolution strategy, which implicitly specifies the geometry of the manifold (Fig. 2(a)) that we subsequently learn. Here, we choose the joint space configuration, $\mathbf{q}$, that minimizes the absolute sum of joint angles, in a different view it minimizes the distance to a convenience (robot default or minimum strain) pose, $\mathbf{q}_c = (0, 0, 0)$, with a joint weighting. Formally,

$$\min \|w\mathbf{q} - \mathbf{q_c}\|^2,\ \textit{subject to}\ f(\mathbf{q}) - \mathbf{x} = 0,$$

where w is a weighting vector, f is the forward kinematics and $\mathbf{x}$ is the goal endpoint position on the plane. We set $w = (4, 2, 1)$, which means that the cost of the first joint offset will be four times as significant as the last joints angle, thus penalizing more its motion.

The resulting $\mathbf{q}$'s trace a smooth nonlinear manifold in joint space, depicted in Fig. 2(a). We note that the manifold



(a) Samples. (b) Learnt manifold. (c) Generated geodesic paths.

Fig. 2. The manifold learning and usage for the 3link arm example. a) Starting with 100 datapoints in joint space, that correspond to task space coordinates as in the inset plot. b) The neighborhood graph in task space (inset plot), and the learnt tangent space that the model predicts for the RBF centers in the high dimensional space. c) Randomly sampled optimal (unconstrained) geodesic paths and corresponding task space trajectories in the inset plot. The thin gray trimesh is a densely sampled reconstruction of the underlying surface, used only for comparison and as a visualization aid.

surface resembles a convex strip that bends backwards towards the edges, much like a section cut of a bent tube. This is the surface that points of the specific optimality criterion trace. Also different redundancy resolution strategies would produce different optimality manifolds. We note that, in general, this kind of information is not explicitly known (in the case of human demonstration) or even visualizable, for many complex problems.

B. Implementation

We start by computing the neighborhood graph of the dataset points. We do this by evaluating the task space distances as the forward kinematics of the system are known. As we require that our set consists of one connected component, we gradually increase the neighborhood distance until no disconnected subsets exist. The resulting neighborhood graph is depicted in Fig. 2(b)(inset plot).

After visualizing the optimality surface we can conclude that the manifold can be naturally represented with a two dimensional tangent space, and we learn a model of $\mathcal{H}_\theta$ with 10 RBFs. We can subsequently evaluate $\mathcal{H}_\theta$ at any point in our joint space. For example Fig. 2(b) shows the tangent basis evaluated at the centers of the RBFs used. Note that the basis vectors are aligned and vary smoothly, i.e. we obtain good generalization within the region of support of the data. This way, in order to *"walk"* on the manifold we need to evaluate the learned tangent basis and follow each *local frame* for each consecutive step, in other words follow the blue and green arrows of Fig. 2(b) for each point in question.

C. Results

Once we have learnt a model of the manifold tangent basis we have access to the geometric properties of the surface. Subsequently the geometry of the manifold can be used to generate optimal geodesic paths. The procedure for generating these unconstrained paths was described in section II-B, and the key advantage is that the generated paths will be of shortest distance and adhere to the manifold geometry. Optimal geodesic paths generated from randomly sampled start and end points are depicted in Fig. 2(c), where the manifold geometry is also plotted (thin gray trimesh) for comparison. Note how the generated paths trace the underlying manifold geometry while also minimizing the deviation from a straight line connecting start and end points (non-geodesic minimum distance). The resulting task space trajectories –the geodesic paths run through forward kinematics– are also displayed in the inset plot at the same

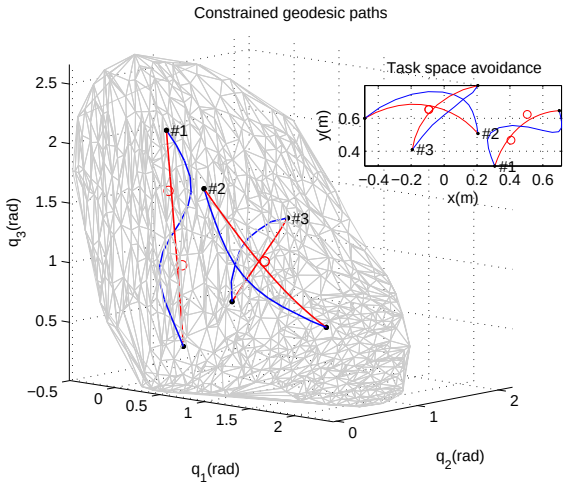
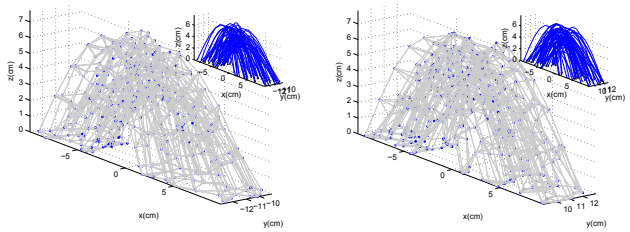


Fig. 3. Example geodesic trajectories that avoid point set obstacles on the learnt manifold. The constrained geodesic trajectories are in blue, and the unconstrained versions are in red.

figure. Note that the resulting task space trajectories are curved, an interesting observation discussed in the next subsection. In most realistic scenarios, we need more control over the geodesic paths that we generate. We would like to be able to specify patches on the manifolds that we would like the generated paths to avoid, while preserving their geodesic properties. This is accomplished by the procedure detailed in section II-C, and results are depicted in Fig. 3. We start with a set of random start and end points and pick a list of obstacle points that intersect the manifold. In Fig.3 the points to be avoided appear as red circles, and effectively trace a patch that can be viewed as a *"no-go"* region on the manifold. The red lines are the predicted geodesic paths that travel through the obstacle regions. The blue lines are the constrained geodesic paths that are optimized with the obstacle patches in consideration. The resulting task space behaviour for this set of examples is visualized in the inset plot of the same picture.

D. Remarks

One interesting observation, regarding the shape of the task space trajectories generated by geodesic paths, is that the shortest path in the 3 dimensional joint space would be a straight line connecting the start and end points. The optimal geodesic paths are the joint space trajectories that connect start and end points and minimize the deviation from a straight line with respect to the manifold geometry.



(a) Right foot training set (b) Left foot training set

Fig. 4. The neighborhood graph, computed for a dataset of 500 points for each swing leg. Both plots show the swing foot midpoint position in task space. The inset plots show the corresponding continuous trajectories in task space.

Now, the manifold is the surface defined as the union of all joint space paths that are optimal with respect to a specific redundancy resolution strategy. These are shortest paths that satisfy the optimality requirements implicitly encoded. In our scenario, the predicted trajectories would be composed of a series of points that minimize the sum of joint angles , thus the task space trajectories would be subsequently optimized with respect to minimum angular change.

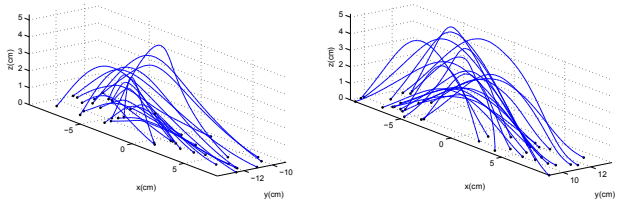
Another point is that the generation of geodesic paths is more efficient – and *much faster* – than numerical optimization as described in section III-A (as also in [2]). One current limitation is that the start and end points used, are assumed to lie on the manifold surface for our experiments. These points are currently generated by the numerical optimization strategy. Once the manifold has been learnt, one could search efficiently for a target point that satisfies the task space goals. In practice, only the goal point has to be searched for and located on the manifold as the start point would already be the current position of the robot.

IV. Experiments on a humanoid robot

The three-link arm experiments are useful for demonstrating the working of the manifold learning and constrained optimal geodesic path planning algorithm. Next, we move to our main example, a humanoid robot. This example allows us to demonstrate an intuitive example of a skill manifold. Our experiments are based on the *Nao* (Fig. 7) humanoid robot, popularly known as the chosen robot for the *Standard Platform League* in *RoboCup*. The *Nao* is approximately half a meter tall and weights $4.3kg$. It has an $500MHz$ *AMD Geode* processor onboard, running embedded Linux. It has 25 DoFs and a variety of sensors. From the point of view of motion synthesis, it is an inherently unstable system, with an elevated center of mass.

We do not have an analytical model of the dynamics of the system. Even if we were to develop an approximate model, it would need to account for varying model parameters, e.g. change in the motor behaviour as the battery gets depleted or motor temperatures vary. Such effects are very hard to capture analytically, although they do matter in practice. So, we prefer to work directly from experimental data. However, we do use a model of the robot kinematics for calculating the relevant foot and pelvic positions in global coordinates.

We focus on the task of walking, with the aim of generating a motion synthesis strategy that achieves full coverage of a reasonably large interval in step length, width and height. In effect, the optimality surface would be the set of all solutions to all possible task space queries, thus a tangent space point would have a local coordinate frame that guides the path in that particular neighborhood. We begin with a redundancy



(a) Generated right foot traj. (b) Generated left foot traj.

Fig. 5. Generated task space trajectories from randomly sampled start and end points. The trajectories correspond to swing foot midpoint trajectories in task space and are stable on the robot.

resolution strategy that would yield walking examples as training data for manifold learning.

A. Training data

We frame the redundancy resolution strategy as a constrained nonlinear optimization problem. Algorithmically, we use a sum of squares (SoS) approach that uses the trust-region-reflective algorithm.

The optimization problem is of the form,

$$\min_{q} \mathcal{J}(\mathbf{q}),\ \mathcal{J} = J^1(q) + \ldots + J^n(q),$$
$$\textit{subject to } f(\mathbf{q}) - \mathbf{x} = 0,$$

where $\mathcal{J}$ is the cost function that is composed of a number of cost factors J^n, f is the forward kinematics and $\mathbf{x}$ is goal task space positions. The cost function is a mixture of task constraints and stability constraints. The cost function we used for data generation evaluates:

- distance between swing foot and goal
- alignment between swing foot and x/y versors
- deviation in pelvis position
- alignment between waist and z versor

The initial pose for the numerical optimization algorithm is a default robot initialization pose with slightly bent knees.

To generate a walking trajectory we start with the desired task space path of the swing leg and the position of the pelvis, and discretize to 10 points. The swing foot trajectories are straight lines from start to goal points while the height of the foot is regulated with a sinusoid with varying apex height. In practice we set the position of the pelvis to be over the support foot and perform a double support weight shift step once the swing leg has reached the goal position. Lastly, we run the optimization procedure described earlier, and get the joint space trajectory of the leg swing and the weight swift phases for each complete task space step path.

The optimization results are approximately constant speed quasi-static trajectories, in the sense that inertial effects are negligible. We collected 50, full body, joint space trajectories for stepping with the right leg and the same amount for stepping with the left leg. Start and goal points of every step have been randomized within a reasonable reaching distance. The inset plots of Fig. 4(b) and 4(a) show the task space trajectories of each swing leg foot midpoint, by running the datasets through the forward kinematics of the system.

B. Implementation

Compared to our previous example, this is a higher dimensional space and sampling is necessarily somewhat sparse. Of the 25 DoFs of the robot, we focus on the 12 DoFs for legs and hips, keeping the arm and head joints at a constant pose. Furthermore we separate each footstep into a swing phase and a weight shift phase. This way we divide the learning process into two components, leg swing

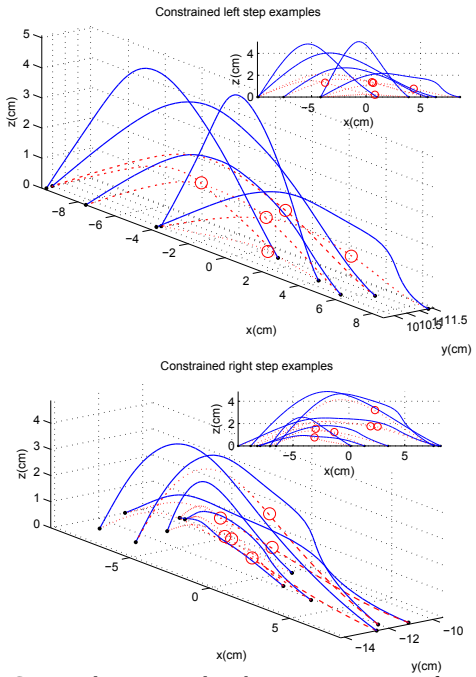


Fig. 6. Generated constrained task space trajectories from randomly sampled start and end points. The red trajectories correspond to the original unconstrained foot midpoint that collide with the obstacle (red circle). The resulting optimized constrained task space trajectories plotted in blue.

manifold and support weight shift manifold - as the measure of optimality is essentially different for each phase.

We begin with the same neighborhood graph computation procedure where we gradually increase the neighborhood distance until the graph is not disconnected (Fig. 4(b) and 4(a)). We set the dimensionality of the manifolds to be 4, with a simple cross validation step that penalizes model complexity while producing stable and reasonable results. In all learnt manifolds we used models with 20 RBFs, and 500 data points that belong to 25 random task space trajectories as described in the previous section.

C. Results

The learnt manifolds are able to produce smooth walking trajectories that satisfy the optimization criteria used to produce the training data. Moreover, trajectories are produced approximately within *one to two seconds*, in contrast to the numerical optimization used to generate the data which required on average approximately *45 seconds* per trajectory, both with reasonable code and on commodity hardware. The computation time of the former increases with the dimensionality of the manifold.

The procedure is able to produce stable walking in the continuum of the reachable space of the robot as depicted in Fig. 5(a) and 5(b) for right and left swings accordingly. One interesting observation is that the robot manufacturer in the accompanying software for walking, specifies that the stepping space of the feet cannot extend more than 9cm. With our manifold trajectory generation we are able to step further and reach stably up to 12cm, nonetheless most of our experimental sampling was constrained to be up to 10cm.

One point to note is that the shape of the generated trajectories in task space is qualitatively different from the training data. The training data is generated by point-by-point kinematic optimization of an artificially imposed sinusoidal sequence of task space points. By fitting the tangent space

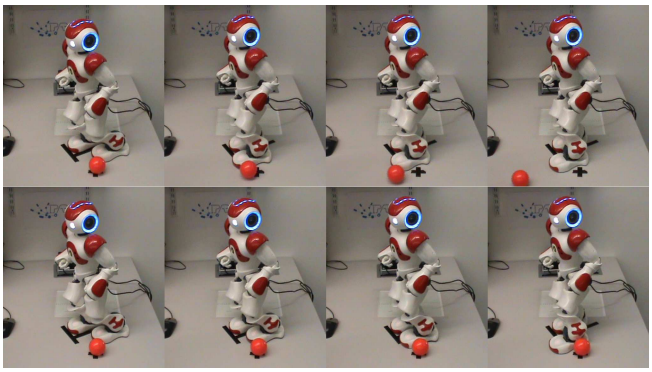


Fig. 7. Nao executing of a planned motion. *Top*; the unconstrained stepping trajectory that hits an obstacle (the ball). *Bottom*; the constrained optimized trajectory where the swing path avoids the obstacle, the ball.

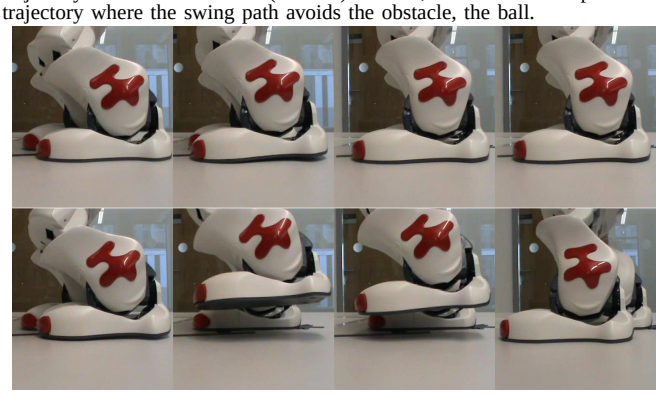


Fig. 8. Detail of a left foot swing. *Top*; the original trajectory that provides minimal foot clearance (approx. 2cm on apex). *Bottom*; adding a obstacle close to the original trajectory pushes the optimization to higher stepping trajectories (here approx. 5cm on apex).

of the manifold to the collection of all such data points, and making all local frames consistent, we extract a manifold that indeed traces the true underlying geometry that the optimization procedure sculpts in the robot joint space.

The above examples correspond to trajectory generation in an unconstrained scenario. As with the 3-link arm example we randomly pick a set of start and end points in task space, generate a trajectory as a geodesic path on the learnt skill manifold and insert an obstacle near the trajectory. Examples of this process are depicted on Fig. 6 for right and left foot midpoint task space trajectories. Note that the dashed red lines correspond to the unconstrained predictions that collide with the perceived obstacles, that appear as red circles.

The efficacy of such an additional degree of control is obvious. To provide a concrete example we have used the constrained geodesic trajectory generation for random obstacle avoidance, staying away from regions in task space that might interfere with the swing trajectory. In the case of going up or down a step, it is often the case that the foot collides with the previous or next step's edge. When such a collision occurs, the robot loses its balance and falls down. Now, we can detect this collision and set this point to be a *"no-go"* point in a point set. In the same state the robot will then skillfully avoid the colliding pose and successfully negotiate the step. The accompanying video provides a number of such examples. Snapshots of such behaviour are also shown in Fig. 7 and 8.

V. Conclusions and Future Work

We present an approach to constrained trajectory generation over a manifold that compactly encodes a continuous family of trajectories corresponding to a robotic skill. This

skill manifold is learnt from demonstration data by approximating the tangent space. Having this manifold along with the local coordinate frames in the form of the tangent space enables efficient ways to handle changing task contexts and obstacles, while respecting intrinsic requirements of the task.

We demonstrated these ideas on two sets of examples - a simulated robotic arm, suitable for visually illustrating core concepts and a humanoid robot behaviour, constrained variable step-length walking. We have demonstrated that the manifolds, defined by solutions to a complex numerical optimization problem, can be learnt from sparse data and that the geometric structure generalizes within and beyond the support of the data. This enables motion synthesis methods where one is able to lazily compute trajectories in response to changing task specifications (perhaps including additions to the underlying cost function expressions) by the constrained geodesics that intrinsically respect the partial specifications that define the essence of the underlying task.

REFERENCES

[1] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[2] I. Havoutis and S. Ramamoorthy, "Geodesic trajectory generation on learnt skill manifolds," *Robotics and Automation, 2010. ICRA 2010. Proceedings 2010 IEEE International Conference on*, 15-19, 2010.

[3] S. Ramamoorthy and B. J. Kuipers, "Trajectory generation for dynamic bipedal walking through qualitative model based manifold learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 359–366, May 2008.

[4] P. Isto and M. Saha, "A slicing connection strategy for constructing prms in high-dimensional cspaces," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1249–1254, May 2006.

[5] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 514–521, 2004.

[6] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009.

[7] M. Stilman, "Task constrained motion planning in robot joint space," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3074–3081, 29 2007-Nov. 2 2007.

[8] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *in WAFR*, 2004, pp. 1–16.

[9] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, 2008.

[10] S. Bitzer, I. Havoutis, and S. Vijayakumar, "Synthesising novel movements through latent space modulation of scalable control policies," in *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 199–209.

[11] S. Calinon and A. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Advanced Robotics*, vol. 23, pp. 2059–2076, 2009.

[12] P. Dollár, V. Rabaud, and S. Belongie, "Non-isometric manifold learning: Analysis and an algorithm," in *ICML*, June 2007.

[13] O. C. Jenkins and M. J. Mataric, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *In International Conference on Machine Learning (ICML)*, 2004, pp. 441–448.

[14] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, August 2001.